

Front-End Development

Labo 3 – CSS 2

J. Pieck, L. Van Houdt, S. Van Peborgh, T. Van Nuland

1 Contents

2 INTRODUCTION 3

3 SETTING UP STRUCTURE 4

4 SELECTOR PLAYGROUND 5

4.1 INSPECT 5

4.2 TYPES OF SELECTORS 6

4.3 MORE SPECIFIC SELECTORS 7

4.4 PSEUDO CLASSES / ELEMENTS 8

4.5 SPECIFICITY OF CONFLICTS 9

BOX MODEL 13

4.6 INSPECT 13

4.7 BOX SIZING 14

4.8 UNITS 17

4.9 MIN/MAX WIDTH 18

4.10 HEIGHT & OVERFLOW 19

4.11 PECULIAR MARGINS 19

4.12 BORDER VERSUS OUTLINE 21

5 NEXT STEPS 22

6 ZIP-FILE INLEVEREN 23

7 REFLECTION 24

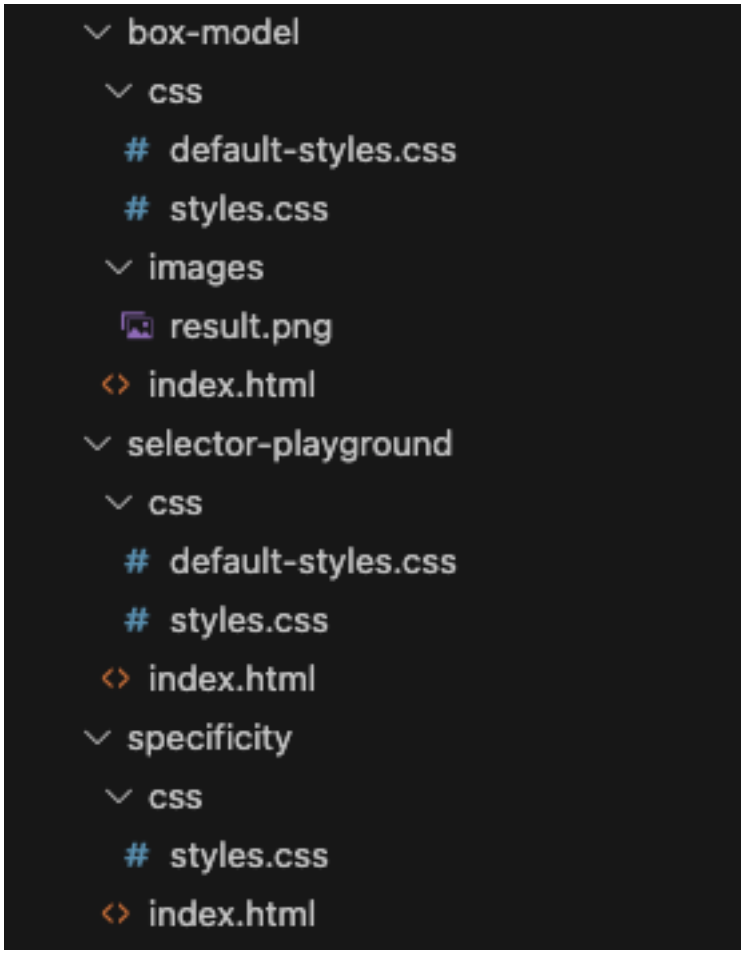
2 introduction

Documentation	Teaching materials on Toledo: slides and code examples
Purpose	Learn to apply more advanced selectors, understand why some rules "win" (specificity), and learn to work with the box model.
Learning Goals	At the end of this lab you will be able to: • Use different types of selectors. • Apply pseudo-classes and pseudo-elements. • Explain specificity and use it in practice. • Configure and inspect the box model (content, padding, border, margin) in DevTools. • Show the difference between content-box and border-box. • Explain the different width units. • Using min/max width in context of RWD. • Explain different overflow configurations. • Explain how margins work in specific edge situations. • Explaining the difference between border and outline.

3 Setting up structure

1. Create a project folder for this lab, for example `labo3/`.
2. Download and unzip the Toledo launch file in this folder.
3. Rename the folder from `lab-3-start` to `labo3`.
4. Open `labo3/` in vscode.

Your folder structure looks like this:



4 Selector Playground

4.1 Inspect

Study the content `selector-playground/index.html`:



In addition to viewing the source code in VS Code, you can also use the browser's DevTools. This way you can see the web page visually and at the same time the associated source code.



How many different class elements are there?



Which elements have an id selector?

4.2 Types of selectors

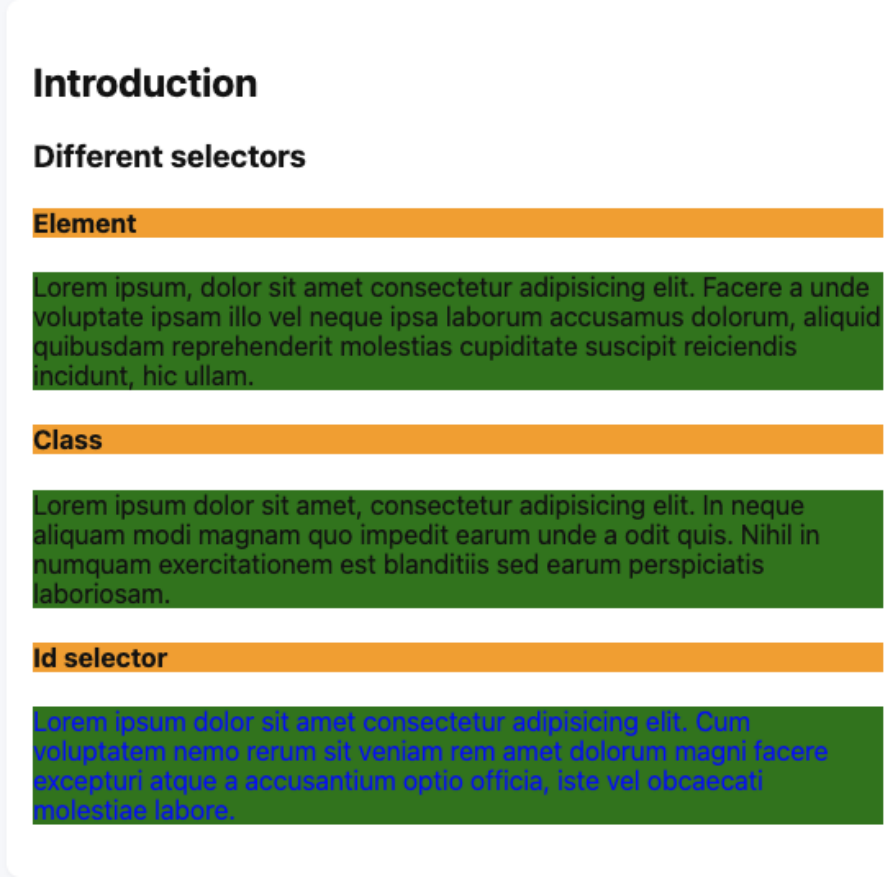
To solve the following exercises, consult the following websites:

- https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_selectors
- https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Styling_basics/Combinators

Open `selector-playground/css/styles.css` in vscode. Write CSS rules that style **only** the requested combinations. Perform the following tasks related to `the introduction` section of the document:

1. Select the `h4` elements and give it an `orange` background color.
2. Select the elements with class `selector-class` and give it a `green` background color.
3. **Select the element with id `selector-id` and make a `blue` text color.**

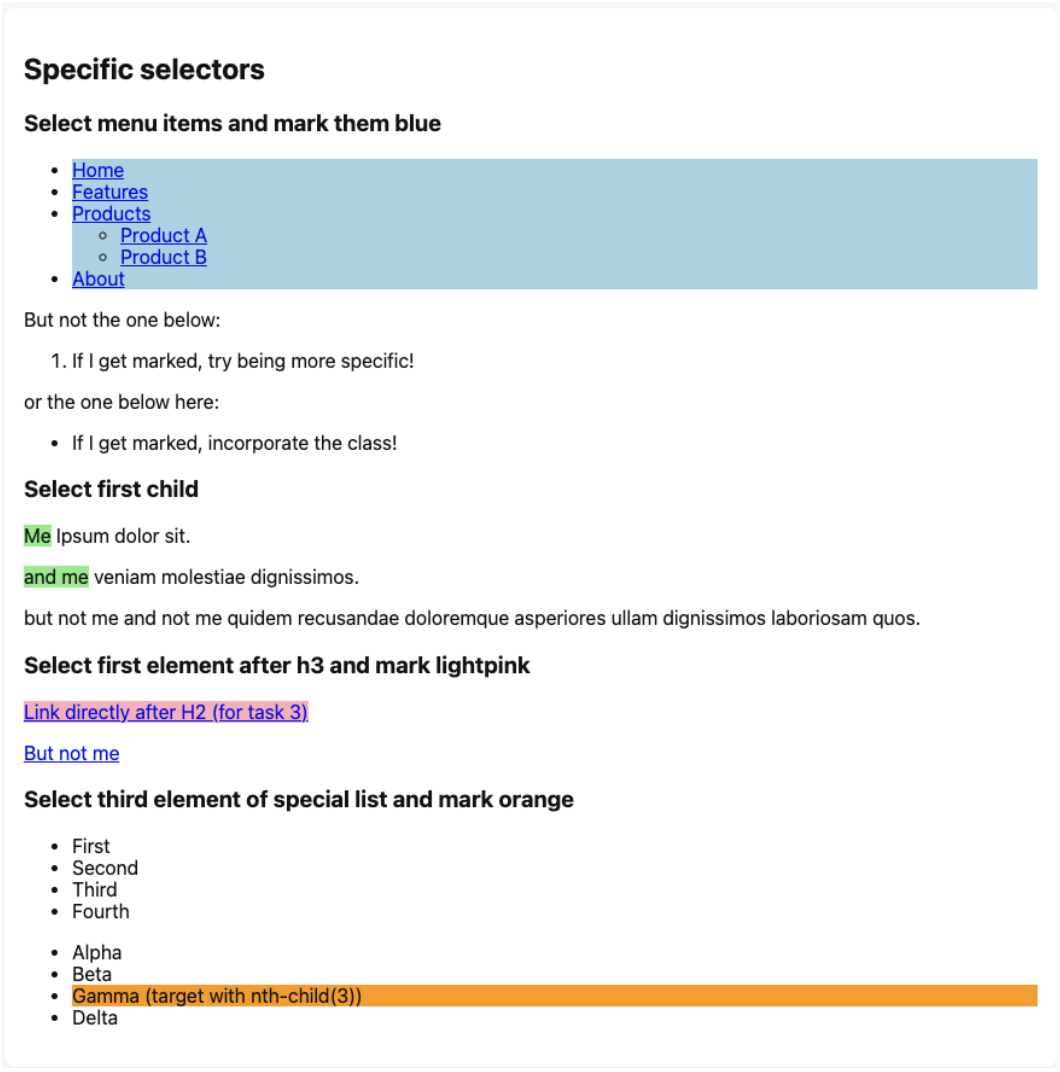
The expected result is as follows:



4.3 More specific selectors

1. **Child combinator**: select all `<li>` that are direct children of `<ul class="menu">` and set the background color to `lightblue`.
2. **Pseudo class**: Select the first `<span>` in each `<p>` and set the background color to `light green`.
3. **Adjacent sibling**: Select the first `<a>` element that immediately follow a `<h3>` and set the background color to `light pink`.
4. **Nth-child**: Select the third `<li>` within the class special-list list and set the background color to `orange`.

The expected result is as follows:



4.4 Pseudo Classes / Elements

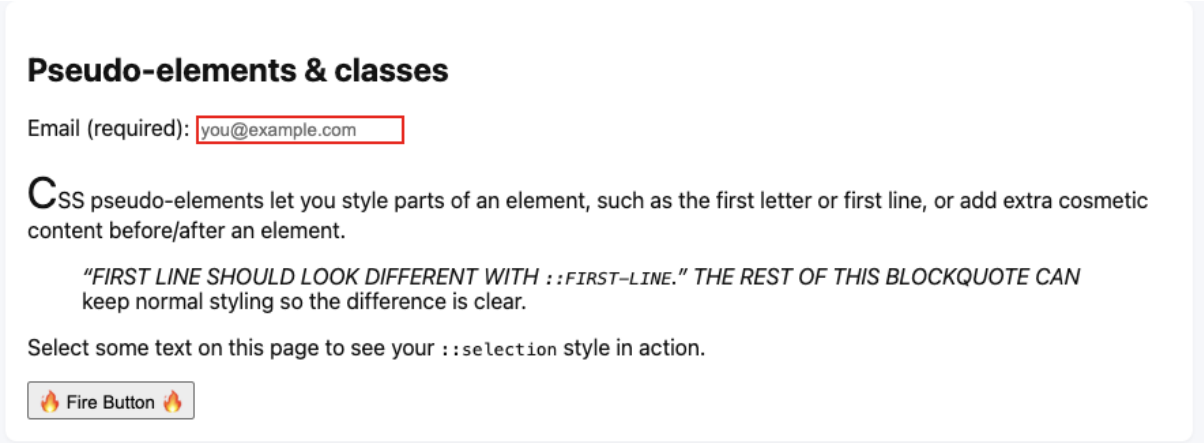
Use the following websites to find the right constructions:

- https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes
- https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-elements

Perform the following tasks:

1. **Pseudo-class :hover**: when you hover `<a>` links, give them an underline and a `light sky blue` background color.
2. **Pseudo-element ::selection**: selected text on the page should be styled with a custom background color.
3. **Pseudo-element ::first-letter**: select the first letter of `dropcap` and increase the text size.
4. **Pseudo-element ::first-line**: style the first line of the `<blockquote>` in the "Pseudo-elements" section uppercase and italic.
5. **Pseudo-elements ::before and ::after**: add a `fire` emoji before and after the `.badge` button.
6. **Pseudo-classes :required, :valid, :invalid**: make the `border` of the email field turn green when it is valid and red when it is invalid. (requires definition of two selectors)

Compare your result with the screenshot below:



4.5 Specificity of conflicts

When multiple CSS rules apply to the same element, the browser must determine which style rule takes precedence. We call this process specificity. Specificity is, as it were, a ranking that indicates how "specific" a selector is. The more specific the selector, the more likely that rule is to win.

Use the following page to learn how to calculate specificity:

- https://web.dev/learn/css/specificity

Use the following page to test whether you understand the subject matter:

- https://specificity.keegan.st/

Open `specify/index.html` in the browser and inspect the source code of the HTML.

The following CSS is configured in `specify/css/styles.css`:

```
p {
  color: blue;
}

.highlight {
  color: green;
}

#intro {
  color: red;
}
```



What color has the paragraph been given?



Calculate the specificity for each configuration, in the form of (0,0,0)?



You can analyse the specificity in vscode in the stylesheet by hovering a concrete selector. For example:

<element class="highlight">

Selector Specificity: (0, 1, 0)

Open `selector-playground/style/styles.css`, add a specific selector so that the text **is always green** without changing the HTML itself.



Do not use `!important` or inline styles to solve specificity problems. While this may seem like an easy solution at times, it doesn't scale with the growth of a project and reduces the maintainability of the website.

The rule of thumb: **the more specific the selector, the stronger wins**. If there are already rules that determine the color (e.g. with a class or id), then you have to write a selector that is more specific.

The expected result is visible below:

Specificity

Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptas tenetur autem dolor alias facere laudantium odio, dolorem unde atque natus eligendi ipsum provident enim illo quos explicabo aliquam quisquam! Repudiandae!

Above paragraph has an id intro and class highlight. We currently support the following styling configuration in `css./styles.css`:

```
p { color: blue }
.highlight { color: green }
#intro { color: red }
```

Add the following CSS code, and solve the questions without viewing the web page:

```
li {
  color: blue;
}
.item {
  color: green;
}
.highlight {
  color: orange;
}
#products {
  color: red;
}
li.item.highlight {
  color: purple;
}
button.btn.special {
  background: yellow;
}
#cta {
  background: pink;
}
```



Predict the colors of:

- The three list items in the `<ul>`
- The first button (`.btn.primary`)
- by second button (`id="cta"`)



Calculate the specificity of each CSS rule in the format (0,0,0]:

li { ...)

.item { ... }

.highlight { ... }

#products { ... }

li.item.highlight { ... }

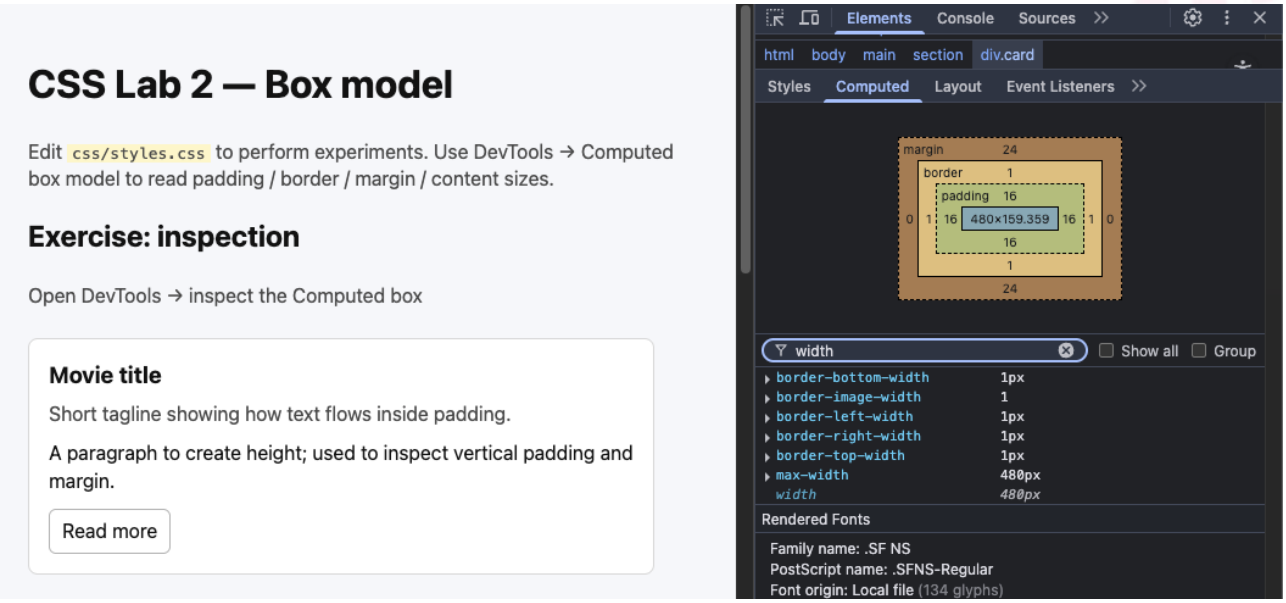
button.btn.special { ... }

#cta { ... }

- Check in the browser to see if your predictions are correct.
- Experiment:
  - Add an extra class to `<li id="products">` in html and write a line `.featured { color: brown; }` in your css. Who wins now?
  - Move the CSS line `#products { ... }` at the very top. Does that change anything?

4.6 Inspect

Open `box-model/index.html` in your browser and open the DevTools, inspect the computed box model of the element with class `card`.



Write down the width, padding, border, and margin of the element with class `card`:

4.7 Box Sizing

Inspect the following html elements:

```
<div class="box box--content">
  <strong>Box A</strong>
  <div class="label">box-sizing: content-box</div>
  <div class="label">
    width: 12.5rem; padding:1.25rem; border:0.3125rem;
  </div>
</div>

<div class="box box--border">
  <strong>Box B</strong>
  <div class="label">box-sizing: border-box</div>
  <div class="label">
    width: 12.5rem; padding:1.25rem; border:0.3125rem;
  </div>
</div>
```

1. Add a CSS selector for:
    a. `.box.box--content` met `box-sizing: content-box`.
    b. `.box.box—border` met `box-sizing: border-box`.
    c. Now view the html with the browser and inspect with the DevTools.



Which of the box elements has the largest width?



Explain in your own words the difference between content-box and border-box, and describe how this choice affects the calculated values in the box model:



Reflect: in which cases would you use a content box? In which cases a border box?

2. Add a CSS selector for:

```
margin: 2rem;

padding: 2rem;
```

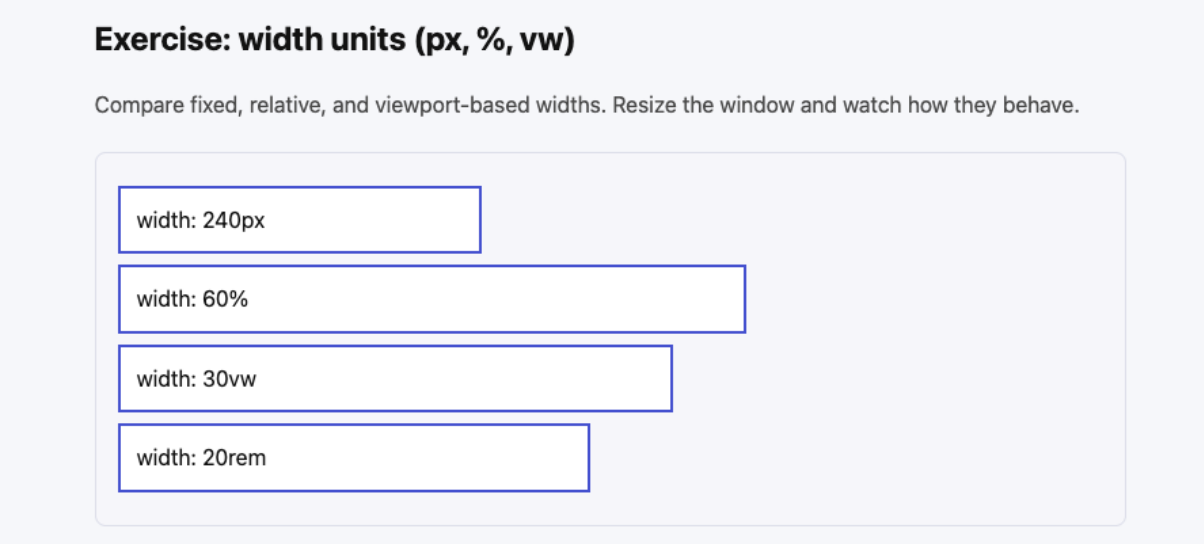


Try out a few different values and explain in your own words what the difference is between margin and padding:



Try adding padding with value: `2rem auto;` Which configuration is equal to this configuration?

4.8 Units



1. Change the size of the browser window. Which box changes in width? Which one remains constant?
2. Of the boxes that changed: which elements were responsible for that change?
3. Change the root font size on html:

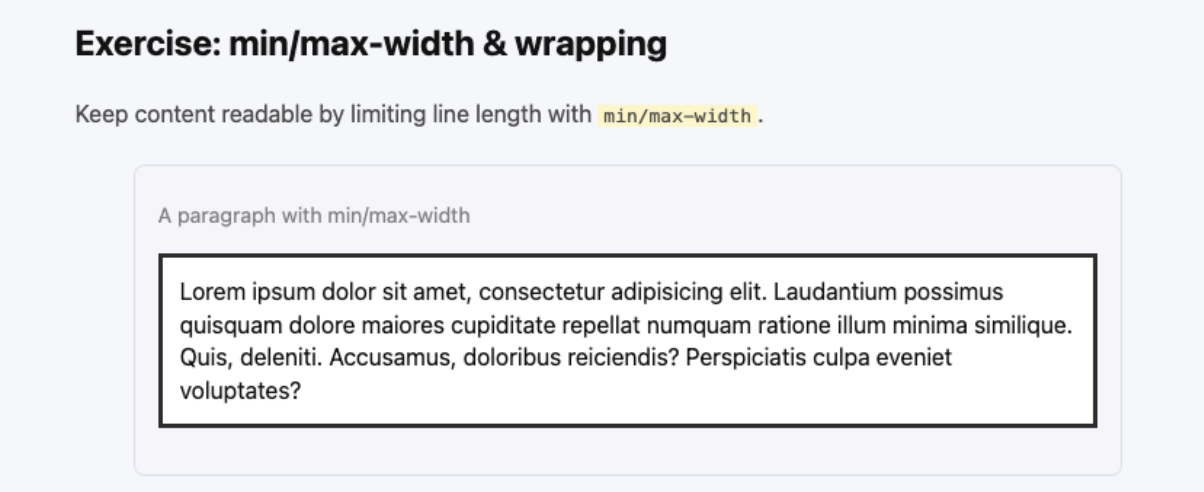
```
html {

font-size: 2em;

}
```

4. Which box changes? Explain the difference.
5. Undo the font size change on the html element.

4.9 Min/max width

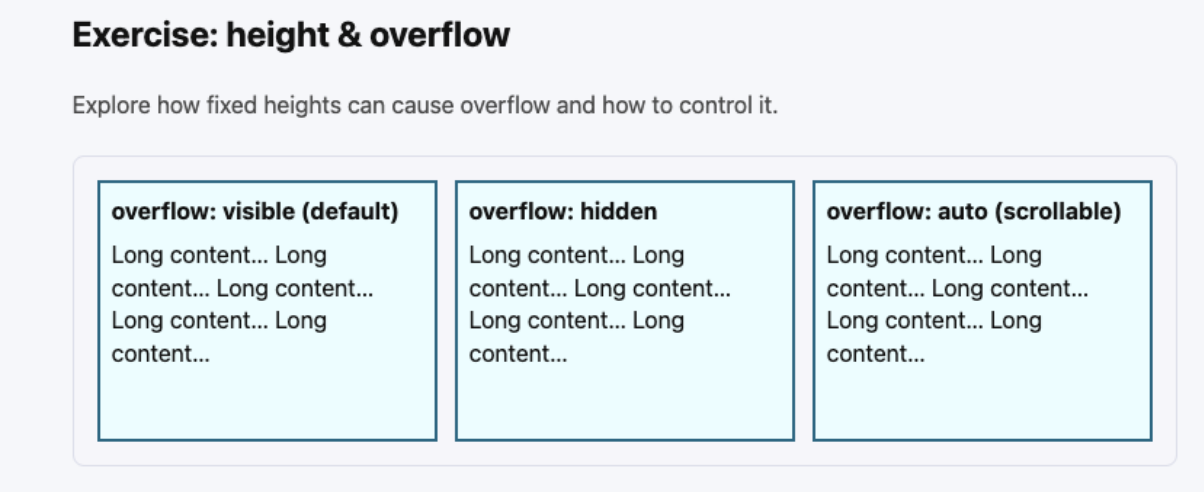


1. `.clamp-card` does not have a width setting.
2. Shrink and enlarge your browser window – what's happening?
3. Set a width of 30rem for `.clamp-card`. Now shrink the browser window to smaller than that value – what did you notice?
4. Change the setting to `max-width: 30rem` (and remove the fixed width from the previous step).
5. Resize the browser window. How does the layout behave differently now?



Reflect: Why is max-width often better for responsive design than a fixed width?

4.10 Height & Overflow



1. Lower the fixed height of `.overflow box` in CSS from `10rem` to `5rem`.
2. Test the user experience of the different overflow settings.

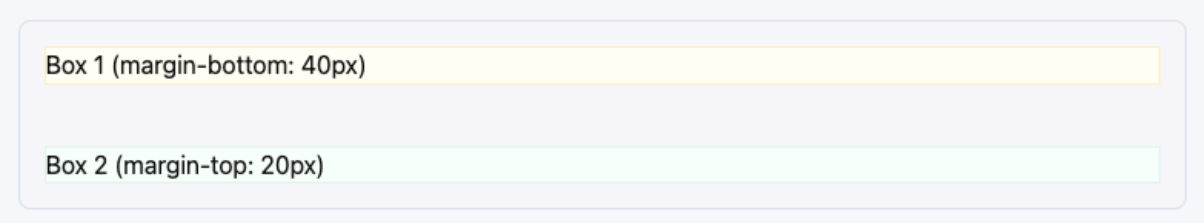
4.11 Peculiar margins



The following examples use pixel as a unit for configuration. This is to make measuring easy. Good practice is to use `rem` units for flexibility and accessibility.

Observe the behavior of vertical margins. Read the short assignments and write down what you see.

Box 1 has set a margin bottom, Box 2 has set a margin top.



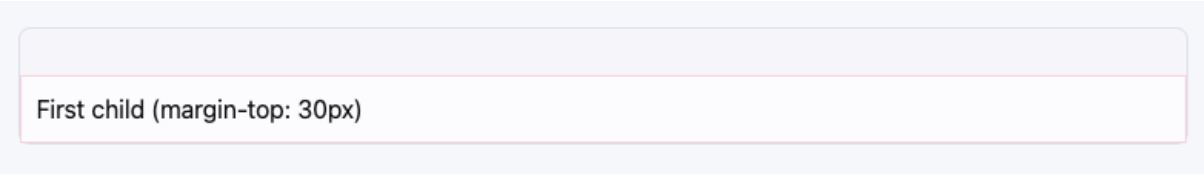


Measure the distance between Box 1 and Box 2. Is the difference 60px or 40px?

When the vertical margins of two elements touch, they merge into one margin instead of being added together. The browser then chooses the largest margin value: if a margin of 20px overlaps with a margin of 40px, then the distance between the two elements becomes 40px.

Parent & child relationship:

Margins can behave unexpectedly in certain edge cases:



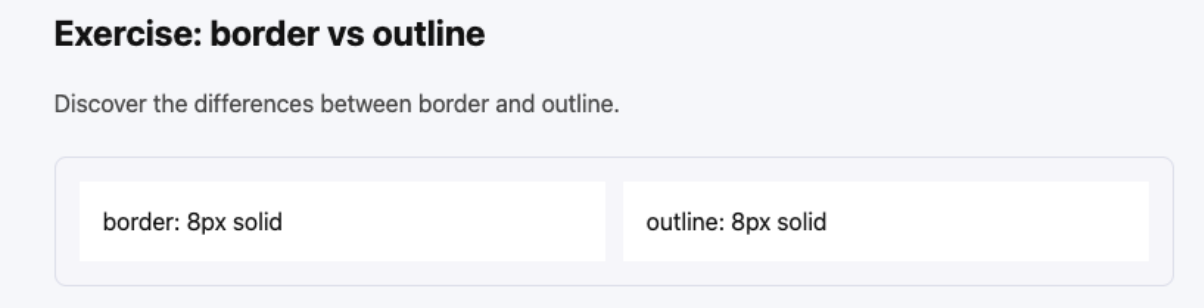
Note that there is no blue background visible although it is configured on the `parent` element. The parent element is located within the margins of the blue border.

Now change the padding from `parent` element to 2px

Do you now see blue background of 2px, 30px or 32px?

When a parent's top has no `border` or `padding` and the first child has a `margin top` , those margins can collapse. See https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_box_model/Mastering_margin_collapsing for more information.

4.12 Border versus Outline



1. Add `border: 8px solid #ef4444;` to `.ovb.ovb--border`.
2. Add `outline: 8px solid #10b981;` to `.ovb.ovb--outline`.
3. Inspect both boxes in DevTools → Box Model.
    a. Which dimensions did the border increase?
    b. What dimensions stayed the same with the outline?
    c. Add `Outline Offset: 8px;` shut. Why doesn't the box model change, even if the outline shifts further away?

Reflect: Why might outline be a better choice than border for focus styles (when a user uses Tab to jump into a form field) or for highlighting elements?

Border is part of the box model and changes the size of the element. Outline is drawn on top of the element, outside the box model, and never changes the layout.

5 Next steps

Experiment with selectors and the box model in your own web page.

Test your knowledge with the following online exercises:

- https://flukeout.github.io/
- https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Styling_basics/Test_your_skills/Selectors
- https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Styling_basics/Test_your_skills/Box_model

6 ZIP-file Inleveren

- Rename the main directory "labo3" as follows: "name_firstname_labo3" e.g. "Doe_John_labo3"
- Through windows explorer (or finder on mac) compress the folder to a zip file.
- No .gzip, .7zip or .rar!
- You should now have a zip file name_firstname_labo3.zip
- You can upload this file to the toledo assignment

7 Reflection



Can you explain when you would use a class, id or combinator?



Do you understand how specificity works and how to resolve conflicts?



Can you give examples of a pseudo-class and a pseudo-element?



Can you explain the difference between content, padding, border and margin?



Do you understand the difference between content-box and border-box?



Can you explain the difference between px, %, em/rem and vw/vh?



Do you understand how vertical margins can merge (margin collapsing)?